ISSN 2079-0023 (print), ISSN 2410-2857 (online)

UDC 004.9:510.635

O. CHEREDNICHENKO, K. V. MELNYK, S. V. KIRKIN, D. V. SOKOLOV, O. M. MATVEIEV

DEVELOPMENT OF AGENT-ORIENTED SOFTWARE COMPONENTS TO RETRIEVE THE MARKETING INFORMATION FROM THE WEB

The article is devoted to researching the processes of extracting marketing information from the Web space. Conclusions are drawn on the need to introduce an information marketing system into modern business activities. A decision has been taken to develop software for the collection and analysis of marketing information. Identified and analyzed the main problems of collecting marketing information in the Web space. External systems for extracting and processing marketing information from the Web space were considered. During the analysis of the subject area, functional and non-functional requirements for the software being developed were formulated. Requirements for the selection of technologies for the development of an information system were defined. The analysis of software development technologies is carried out and the approach to the development of a software component is chosen. Such approaches to software development as: object-oriented programming, service-oriented architecture, component-oriented programming, agent-oriented programming were analyzed. A decision has been made to use the agent three-tier architecture in software development. The most commonly used programming languages in programming systems were: Java, KIF, KQML, AgentSpeak, April, TeleScript, Tcl / Tk, Oz. Analyzed such popular agent platforms and their functions as: JADE, Cougaar, ZEUS, Jason. For the development of software, the JADE platform was chosen, its classes, methods and interfaces were examined. The advantages and peculiarities of the SOLID principle are analyzed. In detail, the levels of the CLEAN architecture are examined. And also explained the possibilities of software implementation of this architecture. A software architecture was developed for the data collection system. In accordance with the requirements, a selection of software development tools has been made. It was decided to use the programming language Java, Spring Framework, GoF design pattern, the template Dependency Injection, SOLID and CLEAN architectural principles. A software component was developed for marketing information gathering systems, which allows to optimize this process. The limitations and ways to improve the software system are analyzed.

Keywords: architecture, JADE, marketing data, CLEAN, agent, program system, Java

О. Ю. ЧЕРЕДНІЧЕНКО, К. В. МЕЛЬНИК, С. В. КІРКІН, Д. В. СОКОЛОВ, О. М. МАТВЕЕВ

РОЗРОБКА АГЕНТНО-ОРІЄНТОВАНИХ КОМПОНЕНТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЛУЧЕННЯ МАРКЕТИНГОВОЇ ІНФОРМАЦІЇ З WEB

Статтю присвячено питанням дослідження процесів вилучення маркетингової інформації з Web-простору. Зроблено висновки про необхідність введення інформаційної маркетингової системи в сучасну підприємницьку діяльність. Прийнято рішення про розробку програмного забезпечення для збору та аналізу маркетингової інформації. Виявлено та проаналізовано основні проблеми збору маркетингової інформації у Web-просторі. Були розглянуті зовнішні системи по вилученню та обробці маркетингової інформації з Web-простору. В ході аналізу предметної області були сформульовані функціональні і нефункціональні вимоги до розроблюємого програмного забезпечення. Були визначені вимоги до вибору технологій для розробки інформаційної системи. Проведено аналіз технологій розробки програмного забезпечення та обрано підхід до розробки програмного компонента. Були проаналізовані такі підходи до розробки програмного забезпечення як: об’єктно-орієнтоване програмування, сервіс-орієнтована архітектура, компонентно-орієнтоване програмування, агентно-орієнтоване програмування. Прийнято рішення про використання агентної трирівневої архітектури в розробці програмного забезпечення. Були розглянуті найбільш часто використовувані в агентних системах мови програмування: Java, KIF, KQML, AgentSpeak, April, TeleScript, Tcl/Tk, Oz. Проаналізовано такі популярні агентні платформи і їх функції як: JADE, Cougaar, ZEUS, Jason. Для розробки програмного забезпечення була обрана платформа JADE, розглянуті її класи, методи і інтерфейси. Проаналізовано переваги та особливості принципу SOLID. В деталях розглянуті рівні архітектури CLEAN. А також зроблені пояснення можливостей програмної реалізації цієї архітектури. Була розроблена програмна архітектура для системи зі збору даних. Відповідно до вимог зроблений вибір інструментів розробки програмного продукту. Прийнято рішення про використання мови програмування Java, Spring Framework, GoF патерну проектування, шаблону Dependency Injection, SOLID і CLEAN архітектурних принципів. Був розроблений програмний компонент для систем збору маркетингової інформації, що дозволяє оптимізувати цей процес. Проаналізовано обмеження і шляхи поліпшення програмної системи

Ключові слова: архітектура, JADE, маркетингові дані, CLEAN, агент, програмна система, Java

О. Ю. ЧЕРЕДНИЧЕНКО, К. В. МЕЛЬНИК, С. В. КИРКИН, В. В. СОКОЛОВ, А. Н. МАТВЕЕВ

РАЗРАБОТКА АГЕНТНО-ОРИЕНТИРОВАННЫХ КОМПОНЕНТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ИЗВЛЕЧЕНИЯ МАРКЕТИНГОВОЙ ИНФОРМАЦИИ ИЗ WEB

Статья посвящена вопросам исследования процессов извлечения маркетинговой информации из Web-пространства. Сделаны выводы о необходимости введения информационной маркетинговой системы в современную предпринимательскую деятельность. Принято решение о разработке программного обеспечения для сбора и анализа маркетинговой информации. Выявлены и проанализированы основные проблемы сбора маркетинговой информации в Web-пространстве. Были рассмотрены внешние системы по извлечению и обработке маркетинговой информации из Web-пространства. В ходе анализа предметной области были сформулированы функциональные и нефункциональные требования к разрабатываемому программному обеспечению. Были определены требования к выбору технологий для разработки информационной системы. Проведен анализ технологий разработки программного обеспечения и выбран подход к разработке программного компонента. Были проанализированы такие подходы к разработке программного обеспечения как: объектно-ориентированное программирование, сервис-ориентированная архитектура, компонентно-ориентированное программирование, агентно-ориентированное программирование. Принято решение об использовании агентной трехуровневой архитектуры в разработке программного обеспечения. Были рассмотрены наиболее часто используемые в агентных системах языки программирования: Java, KIF, KQML, AgentSpeak, April, TeleScript, Tcl/Tk, Oz. Проанализированы такие популярные агентные платформы и их функции как: JADE, Cougaar, ZEUS, Jason. Для разработки программного обеспечения была выбрана платформа JADE, рассмотрены ее классы, методы и интерфейсы. Проанализированы преимущества и особенности принципа SOLID. В деталях рассмотрены уровни архитектуры CLEAN. А также сделаны пояснения возможностей программной реализации этой архитектуры. Была разработана программная архитектура для системы по сбору данных. В соответствии с требованиями произведен выбор инструментов разработки программного продукта. Принято решение об использовании языка программирования Java, Spring Framework, GoF паттерна проектирования, шаблона Dependency Injection, SOLID и CLEAN архитектурных принципов. Был разработан программный компонент для систем сбора маркетинговой информации, позволяющий

оптимизировать этот процесс. Проанализированы ограничения и пути улучшения программной системы.
Ключевые слова: архитектура, JADE, маркетинговые данные, CLEAN, агент, программная система, Java

Introduction. Successful functioning of any company in the market environment can only be ensured only when accurate, complete and reliable information is available. Exactly such information help companies to determine the consumers` attitude towards the product and company, constantly monitor the environment, coordinate the strategy and evaluate the activity, increase the level of advertising work, support in the taken decisions, confirm their own commercial intuition, and increase the activity efficiency [1].

Information systems in one form or another have been used extensively and for a long time now to support marketing activities at enterprises. Nevertheless, the study of the structure and functionality of marketing information systems remains very relevant in connection with the rapid development of information technologies [2].

The external marketing information systems were considered in this work. This marketing systems class designed for regular collection of relevant information. The system of external information is focused on sources and methodical methods, by means of which it is possible to receive information about events and situations from the external marketing environment. The collection of external information involves the accumulation of various structured data:

- about the situation in different markets, especially those where the enterprise works or is going to work;
- forces that operate on the market (existing and potential competitors, consumers, contact audiences, etc.);
- state and trends of the development of macro-factors.

The result of this work is the development of components of the system for collecting, completing, checking the relevance and storage of marketing information about smartphones in real time.

Problem statement. Nowadays, marketing information is of great value. In addition, this value is constantly growing. On the one hand, such growth is due to the transition from local marketing, limited by the state boundaries, to global. On the other hand, with the development of markets and the improvement of technologies, consumers get all the great opportunities in choosing the most satisfying their needs products and services. Modern information technologies help to solve these problems [3, 4].

The ultimate goal of the project is to develop a system for collecting smartphone models from the trading platform that are in high demand (top selling). The following requirements have to be taken into account:

- functionality should not depend on the trading platform;
- the assessment of sales takes place on the basis of the information available on the site of the trading platform;
- the system should have the ability to adapt, study or ability to select the algorithm for determining the top.

Figures 1-2 depict a functional requirements diagram in the IDEF-0 notation (context level diagram and its decomposition respectively).

It is known that the Web is a great repository of information and services of any direction and purpose. At the same time, such a feature hurts Web technologies.

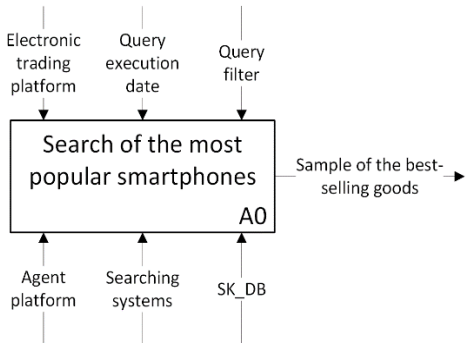

Fig. 1. IDEF0 context level diagram of the functional requirements

If you need to find the exact information, a large amount of time is spent for searching and viewing of a large number of Web-pages. Thus, the purpose of this work is to optimize the search of marketing information, by developing a software component for the system of marketing information collecting.

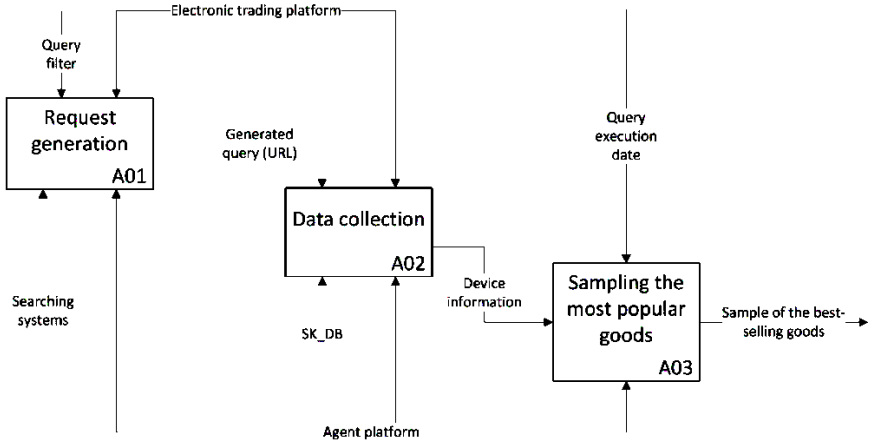

Fig. 2. IDEF0 decomposed diagram of the functional requirements

If you need to find the exact information, a large amount of time is spent for searching and viewing of a large number of Web-pages. Thus, the purpose of this work is to optimize the search of marketing information, by developing a software component for the system of marketing information collecting.

Determine of software development technology. During the process of subject area analysis, functional requirements and limitations of the software system were selected. Consequently, the system requirements for the information system include the following ones:

- the ability to transfer;
- scalability;
- the ability to be modified;
- reliability;
- efficiency.

The above requirements determine the choice of technologies for information system implementation. Among the possible options were analyzed object-oriented (OOP), service-oriented (SOA), component-oriented (COP) and agent-oriented (AOP) programming paradigms (table 1).

It is proposed to use agent architecture when designing in this work. The agent is defined as a computer program that can reasonably act on behalf of the user (another program) to perform the task. Agents, like people, work together so that their aggregate can combine efforts to achieve the goal. As the agent platform (Agent Environment) is usually understood a set of application programming interfaces (API), which provides creation, life cycle, messaging, communication and access to information and agents knowledge bases [5]. Agent platform is assigned functions of the agent environment.

It is worth noting that there are currently no programming languages or development tools that fully meet the needs for agents building. Such a system would have to meet such requirements: ensuring the migration of code to various platforms, availability on many platforms, support for network interaction, multithreading and other [6]. Most commonly in agent technologies used: universal programming languages (Java); "knowledge-oriented" languages, such as knowledge representation languages (KIF); negotiation and knowledge sharing languages (KQML, AgentSpeak, April), agent specifications languages; specialized programming languages for agents (TeleScript); scripting languages (Tcl/Tk); symbolic languages and logical programming languages (Oz) [6].

Today, there are numerous research groups and commercial entities that study agents technology and implement various agents and agent platforms. Due to various approaches to the research of the agent's technology there was necessary to establish a unified standardization organization, the fund of intellectual physical agents.

There are many toolkits for designing MAC, and the most popular among them are agent platforms. The main functions of the agent platform are:

- an environment for the existence and interaction of agents;
- implement certain standards to ensure interoperability and compatibility with other platforms.

Table 2 shows the comparative characteristics of the most well-known agent platforms.

Table 1 – Comparison of implementation technologies

Selection criterion	OOP	SOA	COP	AOP
Main unit	Object	Service	Component	Agent
Degree of autonomy	Within its state due to encapsulation	Not autonomous	Partial autonomy	Complete autonomy and control of its own actions according to its purpose
Flexibility of behavior	Some degree of reactivity	Passive before the call	Some degree of reactivity	Reactivity and proactivity
Interaction with the environment	Limited possibilities	Limited possibilities	Ability to transfer	Mobility, transferability
Interaction with other software components	Mechanisms RPC, RMI	Mechanisms SOAP	Mechanisms of meta-object protocols	Communication on the basis of FIPA messaging
Suitability for solving intellectual problems	Limited possibilities	Limited possibilities	Limited possibilities	Due to the implementation of the model and the ability to study

Table 2 – Comparative characteristics of agent platforms

Comparison criterion	JADE	Cougaar	ZEUS	Jason
Programming language	Java	Java	Visual code generators	AgentSpeak
FIPA-compatibility	+	–	+	–
License for use	Not required	Not required	Required	Not required
Scope of application	Distributed systems that consist of many components	Distributed systems that consist of many components	Basis for writing rules and scenarios	Distributed systems that consist of many components

To implement the information system, the JADE platform is used. Architecture of the system platform JADE is shown in figure 3. The main advantages of this platform are the ability to integrate with other systems, support FIPA-2000 specification and free distribution [7]. The environment consists of two main parts: the actual FIPA-compatible agent platform and Java development agents. The JADE environment is written in Java and consists of Java library classes (packages) that provide application developers with ready functionality fragments and abstract interfaces. JADE comes with a suite of tools that simplify administration and development [8].

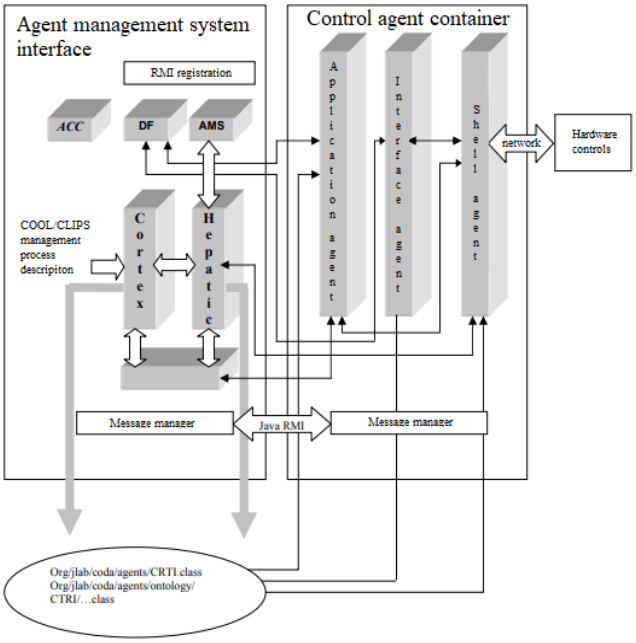

Fig. 3. Architecture of the agent platform JADE

The following main classes are used to create MAC: Agent and Behavior. The Agent class is a common base class for agents specified by the user. From the standpoint of the developer, the JADE agent is a typical instance of the Java class that extends the base class Agent. This implies the inheritance of properties for the implementation of the main interactions with the agent platform (registration, configuration, remote management, etc.) and the basic set of methods that can be called to implement the agent's behavior [8]. The agent's computing model is multitasking and parallel, in which the task (or behavior) is performed simultaneously. Each functionality and/or service provided by the agent must be implemented as one or more behaviors. An internal planner, hidden from the developer, automatically manages the planning of behavior.

The developer determines the agent's actions by specifying its behavior. An agent is able to perform several parallel tasks in response to various external events. In order to be effective in managing the agent, each JADE agent consists of a separate flow of execution and all its tasks (intentions) must be implemented as objects of the Behavior class [8]. The developer when assigning a task to the agent must determine one or more derived classes from the base class Behavior and add some behavior to the list of its tasks. The Agent class provides two methods: addBehaviour() and removeBehavior(), which allow you to control the queue of the agent's tasks, namely add or remove behavior. The handling of the agent can be added whenever necessary, and not only within the Agent.setup() method. The scheduler, implemented by the main class Agent and "hidden" from the programmer, performs a cyclic planning policy among all behaviors that are available in execution queue. Behavior can be blocked, and the agent expects to receive a message [9].

Features of software implementation of the system. The Java programming language, Spring Framework, GoF designing patterns, Dependency Injection template, SOLID and CLEAN Architecture principles [10, 11] were used to develop the software system.

Spring Framework is an open source framework and a container with support of inversion management for the Java platform.

SOLID is an abbreviation of the first letters of the five basic principles of object-oriented programming and design proposed by Robert Martin:

1 The Single Responsibility Principle (SRP) is an important principle of object-oriented programming, which means that a class must be created to perform only one task that it must completely encapsulate. Consequently, all services in this class must be completely subordinated to its implementation. The result of this concept is that there is only one reason for changing the class, which makes it much "healthier" [11].

2 The Open/Closed Principle (OCP) principle is an important principle of object-oriented programming, which means that "program entities, such as classes, modules, functions, methods, etc. must be opened for expansion and closed for changes". This means that they can provide the ability to change their behavior without or with minimal code changes [10].

3 Liskov Substitution Principle (LSP) in object oriented programming is a special definition of a subtype proposed by Barbara Liskov in a 1987 conference in a report "Data abstraction and hierarchy." In the article, Liskov formulated the principle so: if S is a subtype of T, then objects of type S without any changes of the desired properties of this program can replace objects of type T in the program [16].

4 Principle of interface separation. This principle is similar to the principle of a single obligation. The application of this principle consists in the division of too "thick" interfaces into smaller and more specific ones, so that their clients know only those methods that are necessary for their work. As a result, when changing a certain functionality, those classes that do not use it should remain unchanged. That is, the implementation of this principle helps the system remain flexible when making changes to it and remain easy to refactor [12, 13].

5 Principle of dependencies inversion. The principle is formulated as following:

- higher-level modules should not depend on lower-level modules, both types of modules must depend on abstractions;

- abstractions should not depend on the details of the implementation, the details of the implementation must depend on the abstractions.

The dependencies inversion principle solves the problems of unsuccessful designing of programs [12].

On the basis of these principles, Robert Martin presented "The Clean Architecture", which consists of dividing the system into 3 levels:

1 Data layer – the level of data in its pure form, consisting of essences, which are the basic business rules of the system (this level can be both object with methods, and simple set of data structures and functions) [12]. The data layer will include POJOs and get data from cloud or local storage.

2 Domain layer – the level of the business logic of the application that is responsible for the main functionality of the system. Domain layer is responsible for interaction between data and presentation layers by means of interface and interactors. The objective is to make the domain layer independent of anything, so the business logic can be tested without any dependency to external components. Behavior of the domain layer and its` rules are related to a particular application [13].

3 Presentation layer – the level of the user interface, the display of data, the custom events processing (this level converts data from the previous level into a format that is adapted to display. Presentation layer will include normal activities and fragments, which will only handle rendering views and will follow MVP pattern.

The main advantages of this approach are:

- independence from frameworks and specific libraries;
- ease of testing – business rules can be tested separately, without a user interface, databases, etc.;
- user interface independence – the display can be changed without affecting other components of the system, which reduces the chances of potential errors;
- independence from the platform; business rules do not know where they will be used, and not tied to the features of a particular platform;
- database independence – the business logic of the application is not tied to a specific database, which makes it possible to change it at any time without affecting other components of the system.

The last advantage can be realized with the help of the template Repository. This approach is used to separate the logic that receives data, and transforms it into entity models for further work at the domain level. Dependency Injection template is used to use pure architecture and dependency inversion principle to ensure full control of system components. The work of the framework that provides work for such pattern is described by an application that, regardless of the execution, is executed within the dependencies inversion container provided by the framework. Some objects are still created in the usual way using the programming language, but the rest is provided by the container based on a predefined configuration. Thus, this approach makes it possible to substitute the dependencies of system objects using such container without the need of changing of other system components. Current design is component-independent platform and could be tested after each integration is provided. [14].

During the development process, it was concluded that the application of the micro-service architecture and the individual modules creation of the system as independent programs, the execution of which will be processed by the agent platform, that is, the agent behavior (Behavior) will contain the business logic of the individual component (service) of the system. Thus, with the help of aspect-oriented programming, implementation of methods and the processing of their results were provided to agents that use the ACL messaging environment to interact with the services. The system architecture with the use of individual layers is shown in figures 4, 5.

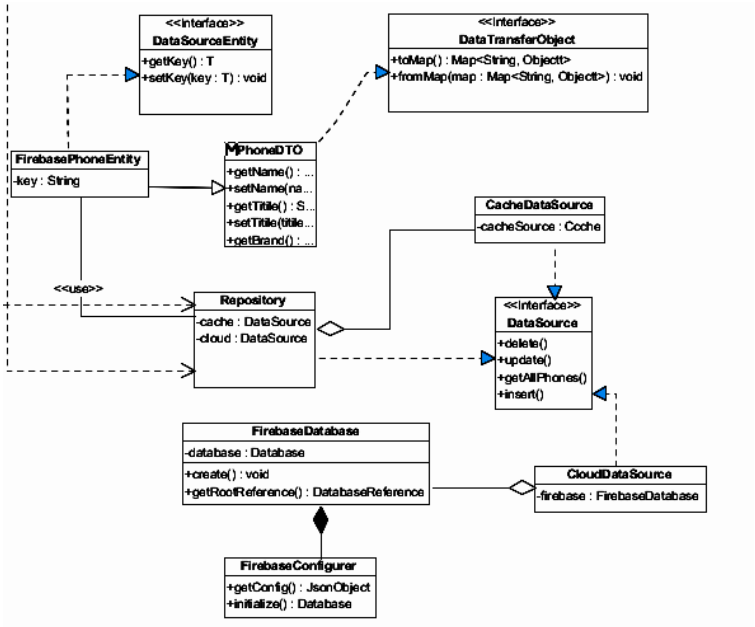

Fig. 4. Data layer

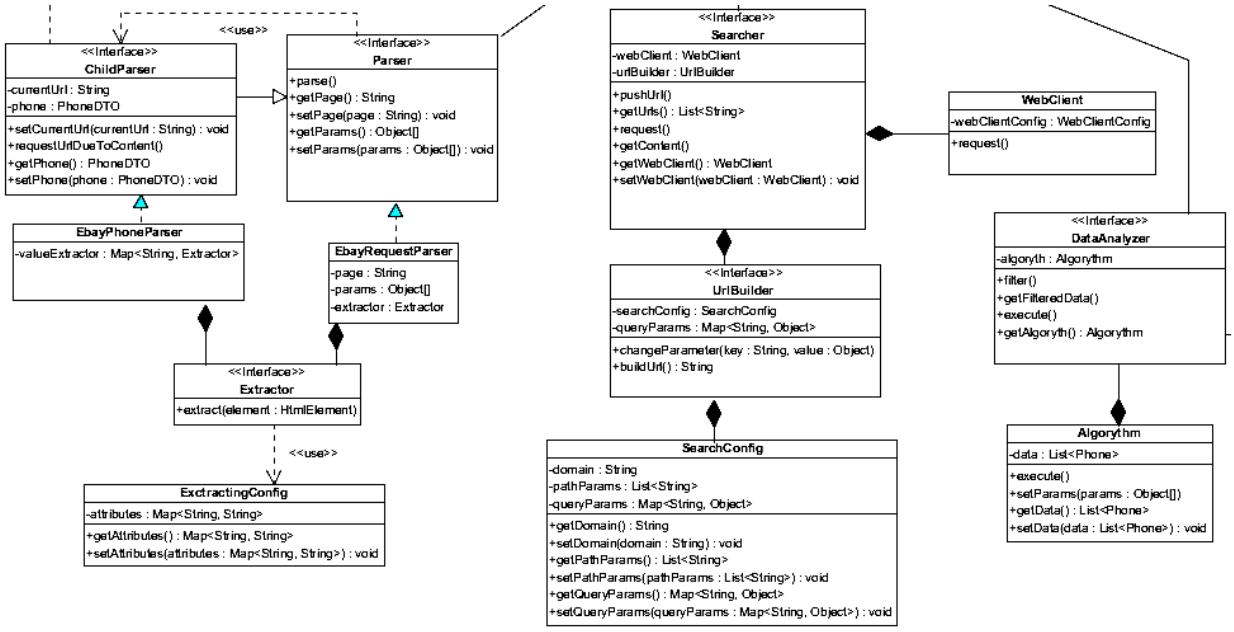

Fig. 5. Domain layer

In the general diagram, we can see that the designed system has 3 layers and 3 microservices, namely:

1 Data layer, which includes classes and data transfer objects interfaces, classes of essence representation of the non-relational database and a Repository design pattern, which consists of two data sources – the cache of the system and the non-relational database Google Firebase, which works with cloud storage services. The data layer contain workflow to manage data in datasource. To do that it is used repository pattern to make CRUD functions convenient and datasource-independent.

The simplest approach, especially with an existing system, is to create a new repository implementation for each business object needed to store or to retrieve data from persistence layer.

2 The domain application layer consists of four microservices:

- agent platform;
- data seeker;
- data analyzer;
- parser and data collector from web-pages.

Service "Agent platform" is responsible for system business process execution within the body of individual agents.

The "Data finder" service performs the process of locating the required goods in the trading venues by replacing the HTTP request parameters, configuration the filters, and transitioning to the data delivery pages.

The "Parser" service analyze an HTML document that is a response for HTTP request from a searcher and collects the necessary data from the product description page.

The "Data Analyzer" service is responsible for analyzing the sample collected by the parser, based on which the search parameters and stored products data will be reformed.

3 The presentation layer logic will be implemented on the client side of the system, namely on a mobile client based on the Android operating system using the MVP pattern (Model-View-Presenter). The connection between servers and clients is implemented with the help of the micro-service of the cloud-based storage Google Firebase Firestore, to which requests are performed both by servers and clients of the system. The layer was designed using micro-service approach to create test-flow for each component.

3 The presentation layer logic will be implemented on the client side of the system, namely on a mobile client based on the Android operating system using the MVP pattern (Model-View-Presenter). The connection between servers and clients is implemented with the help of the micro-service of the cloud-based storage Google Firebase Firestore, to which requests are performed both by servers and clients of the system. The layer was designed using micro-service approach to create test-flow for each component.

Presentation layer has 4 separate components such as:

- data searcher – a search can look for results inside a static local source;
- data analyzer – uses AI algorithms to analyze retrieved dataset and post the list of goods which have the biggest proposal;
- data extractor – uses JSOUP parsing component;
- agent platform – invokes main functional use-case of each components in separate agent.

Due to the fact that the system has a clear distribution on data representation layers and is distributed to separate microservices – the testing processes are not labor-consuming. System components obviously do not depend on each other, the input data can be replaced by

mock objects or by test data for integrating and unit testing.

Analysis and ways to improve the system. During the process of developing and testing the system, it was determined that the agent platform JADE has a list of shortcomings. The main drawbacks are:

1 Contradictory to the SOLID principles, namely the dependencies inversion principle, because the class Agent – depends on the behavior contained in the agent.

2 Using of outdated flow interactions tools. For each agent, a separate stream is created, which badly affects the performance of the system, cause streaming methods use a lot of resources.

3 Use of thread synchronization blocking algorithms. Due to the fact that ACL services use blocking of agents` flows to provide reliable information transfer and maintain the agents` status of performing parallel agents, it does not provide the required increase of performance.

4 An outdated and awkward API for agents launching. To run a single agent, the JADE agent environment uses the classes and the reflection package methods. These operations require a lot of computing power and do not have sufficient reliability.

From the above drawbacks, it was concluded that the system needs to replace the JADE agent platform with a more modern and accompanied agent platform.

Another way to improve the system is to implement a common mechanism for collecting and retrieving information on a separate trading platform using the own domain-specified programming language (DSL). This improvement can provide an increase in system scaling and reduce the cost of implementation of a new trading platform.

Conclusions. The introduction of a marketing informative system into modern entrepreneurial activity is an objective necessity determined by the essence of the information society. As you know, today, and in the near future, information is the main source of the formation of the competitiveness of enterprises or organizations for wealth in general. To stop the process of collecting and processing the information means to risk of the existence of a separate entity of the economic system, and therefore to condemn the troubles all those who depend on it.

The main goal of this work was to develop software for collecting and analyzing marketing information. During the writing the paper, the main problems of information gathering were identified and considered, an analysis of existing approaches and components for solving the problem was made.

During the process of studying the domain area, the main functional and non-functional requirements for the system under development were formed, and its reference system architecture was chosen. The choice of tools is justified in accordance with the requirements.

References

1. *Маркетингова інформація. Маркетингове дослідження.* URL: http://pidruchniki.com/12640422/marketing/marketingova_informatsiya_marketingovi_doslidzhennya (дата звернення 15.04.2018).
2. Карягін Ю. О., Тимошенко З. І. *Маркетинг продукту.* URL: http://tourlib.net/books_ukr/karyagin3-3.htm (дата звернення 15.04.2018).
3. *Маркетингова інформаційна система (MIC).* URL: http://pidruchniki.com/1628041460643/marketing/marketingova_informatsiyna_sistema_mis (дата звернення 26.04.2018).
4. *Система збору зовнішньої маркетингової інформації.* URL: http://stud.com.ua/49872/marketing/sistema_zboru_zovnishnoyi_marketingovoyi_informatsiyi#41 (дата звернення 08.05.2018).
5. *Пошукові можливості web-систем.* URL: http://eprints.isofts.kiev.ua/331/1/05_andon.pdf (дата звернення 08.05.2018).
6. *Методичні вказівки: Інтелектуальні агенти.* URL: http://eir.zntu.edu.ua/bitstream/123456789/2178/1/subbotin_methodical_instructions.pdf (дата звернення 08.05.2018).
7. *Дослідження логічних моделей семантики переговорів інтелектуальних агентів в мультиагентних системах.* URL: http://science.donntu.edu.ua/ius/kirgaev/diss/indexu.htm (дата звернення 08.05.2018).
8. *Об'єктно-орієнтоване програмування. Мова Java.* URL: http://www.dut.edu.ua/uploads/l_1216_25218115.pdf (дата звернення 18.05.2018).
9. Гонтар Ю. М., Чередніченко О. Ю., Янголенко О. В., Вовк М. А. Розробка розподіленої системи обробки бізнес-інформації з використанням агентного підходу. *Системи обробки інформації.* Харків: ХУПС, 2016. Вип. 4. С. 137–142.
10. Симоненко О. А., Сова О. Я., Романюк В. А., Уманець Я. Л. Аналіз існуючих агентних платформ для побудови систем управління вузлами мобільних радіомереж класу MANET. *Системи обробки інформації.* Харків: ХУПС, 2014. № 1 (117). С. 200–203.
11. *Spring Framework.* URL: https://wikivisually.com/lang-uk/wiki/spring_framework (дата звернення 18.05.2018).
12. Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design.* Prentice Hall, 2017. 432 p.
13. *The Clean Architecture.* URL: https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html (дата звернення 18.05.2018).
14. Robert C. Martin. *Summary of book "Clean Architecture"*. URL: https://gist.github.com/ygrenzinger/14812a56b9221c9feca0b3621518635b (дата звернення 18.05.2018).

References (transliterated)

1. *Marketyngova informatsiya. Marketynhove doslidzhennya* [Marketing information. Marketing research]. Available at: http://pidruchniki.com/12640422/marketing/marketingova_informatsiya_marketingovi_doslidzhennya (accessed 15.04.2018).
2. Karyagin Y. O., Tymoshenko Z. I. *Marketing produktu* [Product Marketing]. Available at: http://tourlib.net/books_ukr/karyagin3-3.htm (accessed 15.04.2018).
3. *Marketyngova informatsiyna systema (MIS)* [Marketing Information System (MIS)]. Available at: http://pidruchniki.com/1628041460643/marketing/marketingova_informatsiyna_sistema_mis (accessed 26.04.2018).
4. *Systema zboru zovnishnoyi marketynhovoyi informatsiyi* [System of collection of external marketing information]. Available at: http://stud.com.ua/49872/marketing/sistema_zboru_zovnishnoyi_marketingovoyi_informatsiyi#41 (accessed 08.05.2018).
5. *Poshukovi mozhlyvosti web-cystem* [Searching capabilities of web-systems]. Available at: http://eprints.isofts.kiev.ua/331/1/05_andon.pdf (accessed 08.05.2018).
6. *Metodychni vkazivky: Intelektualni agenty* [Guidance: Intelligent agents]. Available at: http://eir.zntu.edu.ua/bitstream/123456789/2178/1/subbotin_methodical_instructions.pdf (accessed 08.05.2018).
7. *Doslidzhennya lohichnyh modeley semantyky perehovoriv intelektualnyh ahentiv v multyahentnyh systemah* [Research of logical semantics negotiations models of intellectual agents in multiagent systems]. Available at: http://science.donntu.edu.ua/ius/kirgaev/diss/indexu.htm (accessed 08.05.2018).
8. *Ob'yektno-oriyentovane prohramuvannya. Mova Java* [Object oriented programming. Java language]. Available at: http://www.dut.edu.ua/uploads/l_1216_25218115.pdf (accessed 18.05.2018).
9. Hontar Y. M., Cherednichenko O. Y., Yanholenko O. V., Vovk M. A. Rozrobka rozpodilenoyi systemy obrobky biznes-

informatsiyi z vykorystannyam ahentnoho pidkhodu [Development of a distributed business information processing system using the agent approach]. *Systemy obrobky informatsiyi* [Information processing systems]. Kharkiv, KhUPS Publ., 2016, issue 4, pp. 137–142.

10. Symonenko O. A., Sova O .Y., Romanyuk V. A., Umanets′ Ya. L. Analiz isnuyuchykh ahentnykh platform dlya pobudovy system upravlinnya vuzlamy mobil′nykh radiomerezh klasu manet [Analysis of the existing agent platforms for building management systems mobile radio nodes class MANET] Systemy obrobky informatsiyi *Systemy obrobky informatsiyi* [Information processing systems]. Kharkiv, KhUPS Publ., 2014, no. 1 (117), pp. 200–203.
11. *Spring Framework.* Available at: https://wikivisually.com/lang-uk/wiki/spring_framework (accessed 18.05.2018).
12. Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design.* Prentice Hall, 2017. 432 p.
13. *The Clean Architecture.* Available at: https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html (accessed 18.05.2018).
14. Robert C. Martin. *Summary of book "Clean Architecture".* Available at: https://gist.github.com/ygrenzinger/14812a56b9221c9feca0b3621518635b (accessed 18.05.2018).

Received 23.05.2018

Відомості про авторів / Сведения об авторах / About the Authors

Чередніченко Ольга Юріївна (***Чередниченко Ольга Юрьевна***, ***Cherednichenko Olga Yuryevna***) – кандидат технічних наук, доцент, Національний технічний університет «Харківський політехнічний інститут», доцент кафедри Програмної інженерії та інформаційних технологій управління; м. Харків, Україна; ORCID: https://orcid.org/0000-0002-9391-5220; e-mail: olha.cherednichenko@gmail.com

Мельник Каріна Володимирівна (***Мельник Карина Владимировна***, ***Melnyk Karina Vladimirovna***) – кандидат технічних наук, Національний технічний університет «Харківський політехнічний інститут», старший викладач кафедри Програмної інженерії та інформаційних технологій управління; м. Харків, Україна; ORCID: https://orcid.org/0000-0001-9642-5414; e-mail: karina.v.melnyk@gmail.com

Кіркін Станіслав Васильович (***Киркин Станислав Васильевич***, ***Kirkin Stanislav Vasylevich***) – Національний технічний університет «Харківський політехнічний інститут», студент; м. Харків, Україна; ORCID: https://orcid.org/000-0002- 8721-4161; e-mail: skirkin@ukr.net

Соколов Дмитро Вітальович (***Соколов Дмитрий Витальевич***, ***Sokolov Dmitry Vitalevich***) – Національний технічний університет «Харківський політехнічний інститут», студент; м. Харків, Україна; ORCID: https://orcid.org/0000-0003-4572-9500; e-mail: dimitreuzsokolov@gmail.com

Матвєєв Олександр Миколайович (***Матвеев Александр Николаевич***, ***Matveev Alexander Nikolaevich***) – Національний технічний університет «Харківський політехнічний інститут», аспірант; м. Харків, Україна; ORCID: https://orcid.org/0000-0001-5907-3771; e-mail: matwey1970@gmail.com